Chapter 1: HUD's Approach to the Year 2000

1. 1. HUD’s Response to the Year 2000 Challenge

The Office of Information Technology (IT) is aggressively attacking HUD’s Year 2000 problem. Steven Yohai, HUD’s former Chief Information Officer, established the Team 2000 Project Office under the System Engineering Group (SEG) in June of 1996 with the charter to minimize exposure to this risk by actively assisting and coordinating Year 2000 renovation efforts throughout HUD.

1. 1. 1. HUD’s Team 2000

HUD’s Team 2000 consists of HUD’s IT technical staff, HUD’s computer system owners, business partners and vendors, and Year 2000 experts from both the government and the private sector. Anyone addressing the Year 2000 problem at HUD is a member of Team 2000, working to ensure that all HUD’s systems and forms can be used successfully in the Year 2000 and beyond.

Team 2000 is also addressing potential Year 2000 problems other than those of IT systems. The Team 2000 Project Office, along with HUD’s building administrators, are working to ensure elevators, building security, energy and other computer chip controlled devices remain functional. These efforts are not specifically included in this Readiness Guide.

1. 1. 2. Roles and Responsibilities

To successfully manage and coordinate the Year 2000 effort, HUD is employing a highly matrixed structure with responsibility distributed among the Team 2000 Project Office, the Systems Engineering Group (SEG), and other critical IT components, such as Computer Services Group (CSG). **Table 1-1** highlights these roles and responsibilities.

Table 1-1: Roles and Responsibilities Matrix of the Team 2000 Project Office and the Office of Information Technology (IT)

✔ Denotes primary responsibility
SEG - System Engineering Group
CSG - Computer Services Group

	Responsible HUD Organization		
Responsibility Area	**Team 2000 Project Office**	**SEG Division**	**Coordinate With:**
COMPUTER PLATFORM Provide Year 2000 compliant development and production environment.	✔		**CSG**
TOOLS Provide tools to facilitate, identify, change, and test date changes in source or databases, and give instructions on how to use them.	✔		**CSG**
PROCEDURES Describe repeatable process to successfully renovate code.	✔		
STANDARDS Provide HUD-wide standards and guidelines for Year 2000 renovation.	✔		
COMMERCIAL OFF-THE-SHELF SOFTWARE DIRECTION Provide guidance and requirements relating to commercial software.	✔		**CSG**
ESTIMATE MAGNITUDE OF PROBLEM Analyze application inventory to estimate cost and level of effort for reporting and analyzing.	✔		
PROGRAM AWARENESS Inform and coordinate program areas with respect to Year 2000 activities.	✔	✔	
APPLICATION PRIORITIZATION Determine business impact, risk, life expectancy, and application time horizon and integrate into triage-like categories.	✔ (HUD-Wide Scope)	✔ (Program-Area Wide Scope)	
APPLICATION SEQUENCING Refine the prioritization ranking determined in triage with technical considerations to improve development efficiencies.	✔ (HUD-Wide Scope)	✔ (Program-Area Wide Scope)	
APPLICATION SCHEDULING Assign development resources to deliver century-safe software by Year 2000 workplan completion date.		✔	**SEG Division Director**
CONFIGURATION MANAGEMENT Follow configuration management policies for Year 2000 renovation to ensure integrity.		✔	
APPLICATION PREPARATION Run code through assessment tool to determine all occurrences of date fields, create/update system documentation.		✔	
APPLICATION ANALYSIS Determine approach (such as windowing and date expansion), develop requirements, workplan, and test plan.		✔	
APPLICATION CODING / RENOVATION Make all Year 2000 corrections to code, develop bridges and conversion programs, convert data.		✔	
APPLICATION TESTING Perform tests to confirm application can operate in current production environment and can operate with dates before, during, and after January 1, 2000.		✔	
APPLICATION CERTIFICATION Perform final review of application.	✔	✔	

✔ Denotes primary responsibility
SEG - System Engineering Group
CSG - Computer Services Group

	Responsible HUD Organization		
Responsibility Area	**Team 2000 Project Office**	**SEG Division**	**Coordinate With:**
YEAR 2000 COORDINATION Provide a single point of contact for all Year 2000 issues.	✔		
RESOURCES AS REQUIRED Identify and arrange for resource acquisition to augment current staff.	✔	✔	
BUDGETING Plan and report all Year 2000 related budget status information for HUD.	✔	✔	
CONGRESSIONAL REPORTING Prepare responses to Congressional and other requests for project status.	✔		

1. 1. 3. The Team 2000 Project Mission at HUD

The mission of Team 2000 is to ensure that HUD's application systems and the operating components of HUD's computer systems are Year 2000 compliant. The CIO Council Subcommittee on Year 2000 has defined phases of the effort for reporting purposes; **table 1-2** highlights those activity phases and objectives.

Those phases are included in HUD's Year 2000 Project:

1. Assessment,
2. Planning,
3. Renovation,
4. Testing and Certification, and
5. Implementation.

Table 1-2: Activity Phases and Objectives Suggested by the CIO Council Subcommittee on Year 2000

Activity Phase	Objectives and Steps
Assessment	Determine the overall scope and scale of impact of the Year 2000 issue to HUD. Determine the magnitude of the problem and provide a strategic view of what needs to be done, when it should be done and how it should be done.
Planning	Develop and manage the detailed work plan for implementing a Year 2000 solution including: • A HUD wide integrated implementation plan (how applications will be clustered and sequenced along a time line reflecting their interdependencies and prerequisites); • Individual application work plans (detailed work plans for each application); • Formulating contingency plans and monitoring their predecessor to identify when they must be initiated to avoid disruption; and • Identification and scheduling of other, related activities.
Renovation	Successfully and efficiently incorporate the actual changes that enable the modules to be Year 2000 compliant. This includes locating non-compliant code, replacing it in accordance with adopted standards and certifying quality through unit testing.
Testing and Certification	Use rigorous procedures to reduce the possibility of catastrophic failure. This includes extensively certifying the quality of the changes, especially as the data is exchanged among data bases, modules and applications, and as the data is interpreted and used across systems and organizational boundaries. The goal is to complete testing and certification by January 31, 1999.
Implementation	Install the Year 2000 compliant version of the application.

An objective throughout all these phases is to promote the sharing of information critical to understanding and responding to the challenge and to appropriately represent HUD's status to various oversight agencies and constituents.

1. 2. HUD's Approach to Year 2000

1. 2. 1. The Assessment Phase

The objective of this phase is to determine the overall scope and scale of impact of the Year 2000 issue to HUD. It determines the magnitude of the problem and provides a strategic view of what needs to be done, when it should be done and how it should be done.

For the assessment phase, we need to develop a very thorough understanding of HUD's application portfolio, from a high level view of application names and platforms to a detailed understanding of the source and object components within a single application.

In accomplishing this we need to:

1. Develop a complete, accurate list of applications in the portfolio;
2. Determine the application life expectancy;
3. Assess application risk; and
4. Perform application impact assessment.

A Year 2000 perspective pertaining to each of the above is provided in the following sections. Additional technical materials can be found in **Chapter 2**.

1. 2. 1. 1. Develop a Complete, Accurate List of Applications in the Portfolio

The inventory list is the basis of all actions and status tracking. This information helps identify shared features, such as platforms and languages, that help in the selection of tools as well as conversion strategies (such as whether to upgrade or abandon a DBMS).

The inventory maintained by the Team 2000 Project Office, with the guidance and assistance of SEG, serves as the basis for Congressional and Office of Management and Budget (OMB) reporting.

This phase begins by developing a complete application inventory, including names and library locations for the following components:

- ➤ Application source code;
- ➤ Object code;
- ➤ Required compilers and languages;
- ➤ Operating system;
- ➤ Utilities;
- ➤ Files;

- Archives;
- Third party software applications, libraries, and software tools;
- Internal system interfaces; and
- External system interfaces (business partners).

HUD used the Inventory of Automated Systems (IAS) as a starting point for this list. We conducted extensive interviews through questionnaires to augment the information and scanned the source code to capture additional information components. STATUS 2000, a Lotus Notes® database developed by ASD and enhanced by Team 2000, is based on the augmented IAS information. Utilizing the shared interactive capabilities of Notes, STATUS 2000 can be directly accessed and updated by application developers. STATUS 2000 is the definitive source of information relating to Year 2000 systems and schedules. See **STATUS 2000 Database Information**, **Document I**, in the **Reference Library** for user information.

1. 2. 1. 2. Determine the Application Life Expectancy

Life expectancy projections (for example, when the application will be retired) are essential for scheduling decisions and resource assignments.

Each application identified in the inventory must have a clear determination as to whether it:

- Will continue into the next century;
- Will be absorbed by another application, by a specific date, before the turn of the century;
- Is being built and will be compliant when it is placed in production;
- Will expire, as of a specific date, before the turn of the century; and
- Will be replaced as of a specific date, before the turn of the century.

Some applications will not need to be renovated because they are being replaced by other applications. It is important to monitor the progress of the system that is to replace the expiring application. If implementation of the replacing system is delayed, it could mean that the system to be replaced will not expire before it encounters a century date problem. We need a contingency that anticipates when renovation of the expiring application must begin, and triggers an alert should the scheduled replacement appear to falter or be delayed beyond this date. **Table 1-3** helps put this in perspective.

Table 1-3: Life Expectancy and Required Tasks

Application Life Expectancy	Required Year 2000 Tasks
• To be active beyond December 31, 1999 and will be renovated.	A detailed plan must be developed for renovation, testing, and certification of the application.
• Scheduled for replacement, • To be absorbed by another application, or • To be terminated prior to January 1, 2000.	A contingency plan must be developed for each system, whether it is being renovated, built compliant or phased out. Each contingency plan must be developed in enough detail to understand how long it will take--from start to finish—to execute that contingency.
• Currently inactive, • Planned, but will not be built after all, or • Not considered of sufficient value to justify renovation for century compliance.	Written confirmation must be obtained from the users of the system to ensure functionality is not required beyond December 31, 1999.
• Planned, but not yet active.	Planned systems currently under development are expected to be Year 2000 compliant. These systems must complete development and be scheduled for testing and certification.

1. 2. 1. 3. Assess Application Risk

Applications need to be studied to see how dates are used, when failure is likely, and how significant the impact of failure would be to business in order to evaluate when the application needs to be renovated.

A technique to manage the amount of work to be renovated is to categorize applications by their business impact and address those that have the most impact first. That way if we run out of time, the impact should be minimized. Program areas can help tremendously by categorizing applications based on their impact on HUD's operations and mission.

Another strategy: eliminate from the renovation list systems that provide only marginal benefit. **Table 1-4** depicts a way to organize risk information.

Table 1-4: Sample Summary of Risk Information

System ID	Program Area	Business Activity	Supporting Application	Estimated Failure Date of Application	Required Renovation Start Date	Risk Estimate
A43C	Single Family Housing	Claims	Single Family Insurance System/Claims Subsystem	01/01/1999	02/01/1997	$5.4 million per year

1. 2. 1. 4. Perform Application Impact Assessment

The impact assessment provides specific system knowledge about where and how dates are used. This knowledge is necessary for cost and staff estimating and as input into the analysis phase of renovation.

If particular care is exercised early in this phase, it can save time and reduce efforts later (for example, when it is discovered at a late date that one of the source modules does not exist).

This may require additional work, but can benefit the application in the long run. Many of these same exacting tasks are required before the application can be placed under automated configuration management controls, such as Endevor on the Hitachi platform, Software Quality Assurance (SQA) on the Unisys platform, and in certain cases, Source Safe or PVCS on the LAN. Steps include:

- Verify the current executable is derived from the current source code; initiate recovery if the correct source code is not found. The application may need to be upgraded because the computer platform on which it runs (such as the operating system or third party software) must be upgraded to be compliant.
- Examine the operating system and third party software to determine if and when upgrades may be introduced if needed or appropriate. The examination will be performed by the Team 2000 Project Office and Computer Services Group.
- Validate estimates from impact assessment tools by using the first five systems to undergo renovations. These systems will also be used to evaluate the effectiveness of Year 2000 standards and procedures.
- Identify the completeness and integrity of the source code and provide statistics and cost estimates by using a high level scan of source components.
- Identify specific date references, cross references and software components useful during analysis through a lower level review of source components.

The automated tools used in impact assessment are platform specific. Under the guidance of Team 2000's Project Office:

- Hitachi source code is scanned using Platinum Technology's SystemVision 2000;
- Unisys software is scanned by proprietary tools administered by Unisys/Data Dimensions Inc. under contract to HUD; and
- LAN software is scanned using techniques and tools developed for us by Unisys/ Automated Business Systems Services.

A project can arrange to have their software rescanned by contacting the Team 2000 Project Office for assistance.

1. 2. 2. The Planning Phase

The objective of this phase is to develop and manage the detailed work plan for implementing a Year 2000 solution.

It includes:

- A HUD-wide integrated implementation plan (how applications will be clustered and sequenced along a time line reflecting their interdependencies and prerequisites);
- Individual application work plans (detailed work plans for each application);
- Formulating contingency plans and monitoring their predecessors to identify when they must be initiated to avoid disruption; and
- Identification and scheduling of other, related activities.

For the planning phase, we need to formulate each system's renovation schedule and HUD's overall renovation schedule. To accomplish this we need to:

1. Develop a detailed work plan for each project;
2. Conform plans to Auditing, OMB, and Congressional reporting requirements;
3. Cluster systems having strong interdependencies together;
4. Identify time constraints based on life expectancy, failure date and prerequisite events;
5. Establish firm testing and implementation dates for all projects in a cluster;
6. Complete a comprehensive Integrated Implementation Plan; and
7. Identify Contingency Plans.

A Year 2000 perspective pertaining to each of the above is provided in the following sections. Additional technical materials can be found in **Section 2.4 (Planning)**.

1. 2. 2. 1. Develop Detailed Workplan for Each Project

The overall workplan for the Year 2000 solution at HUD must be developed from the workplans determined for each individual system. The individual workplan should constitute a complete and reasonably accurate estimate of the work to be performed on that system; an estimate which assures optimum resource utilization, achieves HUD's goals, and assure sufficient coordination and sequencing of work. Steps include:

- Compare inventory statistics to confirm that we are basing our plans on a complete and accurate inventory;
- Confirm that the current production version is derived from the source inventory;

- Consider several compliance approaches:
  - **Do nothing** (the application won't be in use at the turn of the century or its handling of dates is not disruptive),
  - **Modify the system** (using various techniques from data expansion to windowing),
  - **Replace the system** (buy an off-the-shelf package or absorb the functionality into another system),
  - **Retire the system** (With limited time and resources, it will not be cost effective to renovate marginal systems. We should encourage owners to rejustify.),
  - **Re-engineer the system** (If we haven't already started, we probably don't have time to consider this as a serious option.); and
- Determine a conversion strategy for each application. It may need to be revised based upon a broader analysis that considers systems that share the same files and data.
- If external interfaces are being modified, existing contractual agreements with HUD's business partners may require change. Allow sufficient time in the schedule.

1. 2. 2. 2. Conform Plans to Auditing, OMB, Congressional Reporting Requirements

The designated milestones of interest to HUD's Team 2000 Project Office are those needed to comply with Auditing, OMB, and Congressional reporting. Requirements. Develop system workplans accordingly:

- Set and monitor start and end dates for the following phases:
  - Application Analysis,
  - Renovation,
  - Testing,
  - Certification and
  - Implementation.
- Provide weekly updates to the Work Breakdown Structure. For every task started but not completed, verify and revise the *end date*.
- For every project behind schedule by more than one month:
  - Provide an explanation for the delay and a management plan to accelerate the effort,
  - Provide a new schedule, and
  - Describe funding and resources devoted to completing the task.
- For every task behind schedule three or more months:
  - Provide an explanation of why the system remains behind schedule and what actions are being taken to mitigate the situation; and
  - Provide a summary of the contingency plan to perform the business function should the replacement or conversion not be completed on time.

1. 2. 2. 3. Cluster Systems Having Strong Interdependencies Together

Systems that share the same files and data or that participate in performing a single functional transaction or that function as a unit are best dealt with as a coordinated group or cluster, even though multiple projects and development teams may be involved. Steps include:

- ➤ Examine files, databases, copybooks, screens, reports, direct calls and messages to identify dates to be modified and possible inter-application interfaces;
- ➤ Keep clusters small to increase the probability of on-time delivery; large releases strain even the best practices; and
- ➤ Employ bridges to limit the number of synchronized releases required. To reduce bridges and redundant testing, cluster together systems that exchange data.

1. 2. 2. 4. Identify Time Constraints Based on Life Expectancy, Failure Date and Prerequisite Events

Careful identification of time constraints will improve the plan:

- ➤ Eliminate systems that are due to expire; maybe even accelerate their retirement or replacement. Remember to identify and track the replacement early enough that a contingency can be initiated if the replacement isn't available as planned;
- ➤ Determine when the system will first miscalculate or be disrupted because of the century change. This will help determine the sequencing of work;
- ➤ Determine what other events must precede work on an application (for example, must a compliant compiler be installed on the test system?) to make sure the environment is ready when needed;
- ➤ Attempt to implement an application at least six months before it experiences its first failure date;
- ➤ Identify external interfaces that send electronic data files to HUD or receive electronic data files from HUD. External business partners must be identified. The specifications and timeframes must be shared and understood; and
- ➤ Consider when upgrades to compliant third party systems or operating software will be available for development.

Take a pragmatic view of the overall plan to avoid scheduling problems and balance the installation workload.

1. 2. 2. 5. Establish Firm Testing and Implementation Dates for all Projects in a Cluster

Significant interface testing will be especially important to assure the integrity of the data or transaction. By doing string testing of the cluster as a whole, transactions of data among systems are moved through the cluster. At the end, the data is examined to confirm that the century still has integrity.

Allow enough time for testing. Have a formal plan. Regard interface testing and implementation almost as a contractual commitment.

1. 2. 2. 6. Complete a Comprehensive Integrated Implementation Plan

The basis of all OMB and Congressional Progress Reporting is the schedules represented in HUD’s Integrated Implementation Plan. The basis of planning and achievement will be the Integrated Implementation Plan.

Change is unavoidable and ever present. To manage change, a solid framework is needed to track the work and determine the impact of those changes:

- ➤ Group applications into clusters to be implemented together; sequence those clusters along a time line and establish firm schedules understood by every system in the cluster as well as by all those organizations that must support the implementation of each cluster;
- ➤ Logically group systems that are closely related or inter-connected (to minimize bridges and minimize reworking required if they were implemented independently);
- ➤ Identify dependencies that influence when these groups (clusters) can be positioned on a timeline;
- ➤ Schedule the most significant systems, based on risk, prerequisites, etc.;
- ➤ Consider limiting cluster size based on lines of code; extent of date changes; percentage of compliant dates; number of programs and I/O; and
- ➤ Consider testing impact:
  - Try to avoid multiple test passes and repeated tests of the same module; and
  - Limit the need to reconcile data from two different systems.
- ➤ Consider dependencies:
  - Reduce dependence on external vendors and information sources by converting those applications early;
  - Convert systems early that would minimize dependencies on vendor software; and
  - Address difficult problems early, rather than later.
- ➤ Consider opportunities for specialization (such as handling all of one language at one point in time; or a CICS upgrade at the same time);
- ➤ Consider when these clusters are best done to:
  - Address problematic, large or critical processes first,
  - Achieve target goals,
  - Balance human and computer resource demands,
  - Minimize or maximize customer impact, and
  - Precede time horizon to failure by at least 6 months.

1. 2. 2. 7. Identify Contingency Plans

An early warning system should be put in place and a contingency plan readied that provides options should assumptions or plans not work out.

For example, if we assume we do not have to renovate an application because it is being replaced, we need to:

- Identify the replacing system;
- Monitor the status of the replacing system;
- Have a well-formulated plan of what we would do if the replacing system is not delivered in time; and
- Understand how long it will take to implement this plan.
- Start the contingency plan no later than its required start date so that it can be implemented in time, even if this means work is started before it is certain the original plans cannot be achieved.

Table 1-5 depicts a way to organize this contingency information.

Table 1-5: One Way To Organize Contingency

System ID	Expiring Application	Estimated Failure Date	Contingency's Required Start DT	Replacing Application Name	Replacing Application Scheduled Dates	Replacing Application Actual Dates
C49	CIMS	01/01/2000	01/01/1999	CO7	5/1997	On Schedule
F90	MIDLIS	12/31/1997	01/01/1996	F52	12/1997	On Schedule

NOTE: The expiring application will be replaced by the replacing application.

1. 2. 3. The Renovation Phase

The objective of this phase is to successfully and efficiently incorporate the actual changes that enable the modules to be Year 2000 compliant. This includes locating non-compliant code, replacing it in accordance with adopted standards and certifying quality through testing.

For the renovation phase, we need to acknowledge the large amount of change we will be introducing and the meticulous nature of this work in order to be successful.
The following are some perspectives and techniques that can assist with this challenge, categorized into the typical developmental activities:

1. Analysis and design,
2. Documentation,
3. Modifying source code,
4. Database redesign and reorganization,
5. Data conversion,
6. Bridges,
7. Project management, and
8. Configuration management.

A Year 2000 perspective pertaining to each of the above is provided in the following sections. Additional technical materials can be found in **Chapters 2** through **4**.

1. 2. 3. 1. Analysis and Design

Take the time to establish a clear and comprehensive vision to guide every step of the renovation process.

Conform with the following standards:

- HUD's Standards and Guidelines for Year 2000 (See the **Standards for Year 2000 Conversion / Year 2000 Technical Panel**, **Document E** in the **Reference Library**);
- Established rules for century representation (CCYYMMDD);
- Federal Acquisition Regulation (FAR) language;
- Interagency Data Transfers standards (National Institute of Standards and Technology/Federal Information Processing Standards); and
- Memorandum of Agreement between Federal and State Governments on Year 2000 issues, dated December 10, 1997, requiring data exchanges in a 4-digit contiguous year format.

Select an approach for conversion of date fields and determine interface specifications and technique early.

1. 2. 3. 2. Documentation

Documentation requirements will be enforced by audit and Year 2000 record retention requirements. Regular status reporting to OMB and other oversight agencies is mandatory.

In all cases developers will be expected to comply with HUD documentation standards and conform to NIST FIPS PUB 38 documentation requirements for development phases. See **Section 4.3.5** (Documentation) for more information.

It is extremely important that comprehensive documentation procedures are established and followed:

- Any litigation will require reconstruction of actions and basis of decisions;
- As bugs surface unpredictably, emergency recovery—up through and including the Year 2000—will require us to quickly locate Year 2000 fixes that were performed as much as 18-24 months earlier.
- Auditing requirements will include the ability to trace who made what change when and that no other changes could have been introduced.

Consider instituting specific naming conventions:

- Identify bridge as a bridge so it is easy to find and remove later;
- Identify the source system and the object system so it is easy to identify between which applications a bridge exists; and
- Identify the version or generation through the naming convention used. Not only does this make it easy to know which version may be outdated, but the process of naming reinforces the need to be alert.

1. 2. 3. 3. Modifying Source Code

The Year 2000 problem presents an opportunity to replace obsolete technology. Some opportunities will be forced upon us because, for example, utilities are not compliant and will not be maintained by the vendor. Others may be things we had been planning to do but could never get to the top of the priority list.

Automation can assist with more rote activities; but in general, there is no safe way to perform the code replacement without line by line evaluation by the programmer.

Employ strong change control and configuration management procedures to keep track of what is happening. Source code changes must be documented in a highly visible manner. This is particularly important when, as in the Year 2000 effort, an audit is likely. The **Tools Overview**, **Document F** in the **Reference Library** describes tools which can assist in this step.

1. 2. 3. 4. Database Redesign and Reorganization

Be alert for key fields that have special meaning. (For example, 09/09/99, 12/31/99, or “00”). The **Tools Overview**, **Document F** in the **Reference Library** describes tools which can assist in this step.

1. 2. 3. 5. Data Conversion

Consider not only the current active files, but archived data as well.

It may be possible to reduce the level of effort by developing templates because the design of conversion programs tends to be quite simple and similar. Templates could help in the rapid creation of basically similar programs. As part of this process, a list of fields to be converted is needed.

This list, which includes the old and new specifications, should be made available to all the personnel on the team for ready reference.

Don’t forget to include in the application test plan a means to test the data conversion process.

Include within data conversion the effort to appropriately "age" the test data files. As the calendar date is advanced by "x" years, so too the data input stream and other files will most likely need to be aged an equivalent amount.

Determine whether or not there is a need to reflect data in a manner consistent with another data store within HUD. For example, if any sort of data reconciliation takes place, is it possible to make the process easier through data conversion decisions?

Has there been adequate time reflected in the conversion schedule to freeze the application data bases long enough to convert them without the loss of data?

Check the **HUD Year 2000 Tools Overview**, **Document F** in the **Reference Library** for tools, such as File/AID, that might assist.

1. 2. 3. 6. Bridges

Enable non-compliant access by creating a utility (bridge) that reads data in one format and writes it in another format.

Agree to interfaces early, even in advance of renovation. Include in these agreements who builds or eliminates the bridge and under what circumstances each action should be taken.

Use the following guidelines:

- When an application is ready to *receive* compliant data (including newly compliant data), a bridge should be built that accepts the current, non-compliant file and translates it to the newly compliant format. This bridge should remain until all senders are ready to transmit compliant data and it has been confirmed that the data deemed compliant truly *is* compliant.
- When an application is ready to *send* compliant data (including newly compliant data), a bridge should be built that provides a second output by converting the new format back to the non-compliant format. This bridge should remain until all recipients are ready to accept compliant data.
- Team 2000, with the assistance of the development teams and the Computer Services Group, should examine all bridges as of December 31, 1998 to identify those bridges that cannot be decommissioned.

File/AID, a tool that facilitates data conversion on the Hitachi, may be useful in bridge creation. See the **HUD Year 2000 Tools Overview**, **Document F** in the **Reference Library** for more information on this tool.

Institute defensive editing techniques if the risk of receiving bad data from external sources is high. Some examples of defensive editing include:

- ➤ Perform date validations;
- ➤ Check date dependent calculations for reasonableness; and
- ➤ Produce warnings or soft errors if doubtful.

1. 2. 3. 7. Project Management

To anticipate the nature of the work, its components, interdependencies and resource requirements and impediments, both the Team 2000 project office and the individual project leaders will need to plan, monitor and control the achievement of HUD's Year 2000 renovation goal.

Team 2000 will provide an integrated, agency perspective and will perform common functions such as coordinating standards, achieving compliant platforms, mediating and escalating differences, and serving as the focal point for Congressional and OMB oversight reporting. In executing these responsibilities Team 2000 will need:

- ➤ Budget and actual expense information on a regular basis; and
- ➤ Status information regarding schedules and tasks at a detailed level.

1. 2. 3. 7. 1. PARMS (Resource Reporting)

Each organization working on the Year 2000 Project will record their time to categories established in the Project and Resources Management System (PARMS). This will enable management to evaluate if the project is maintaining the drive and momentum necessary to achieve its component objectives. See the **PARMS and Year 2000 Budget Reporting Requirements**, **Document H** of the **Reference Library** for more information.

1. 2. 3. 7. 2. Status and Milestone Reporting

Each task required to deliver Year 2000 compliant applications for HUD must have a detailed work plan. A report on the status of these tasks must be delivered on a weekly basis to the Team 2000 Project Office. There are two parts to this report:

- ➤ A **General** assessment of the status of the project; and
- ➤ **Detailed** tracking of task start and end dates and how they match the plans.

The General Accounting Office has published a checklist against which project management and the agency can assess progress. This checklist is reproduced in the **Year 2000 Checklists, Worksheets, and Templates**, **Document D** in the **Reference Library**.

1. 2. 3. 8. Configuration Management (CM)

Although configuration management is perceived as a very meticulous and time-consuming process, it safeguards the operational integrity of our applications by administering source and object libraries containing all the components of each business application.

Each development team can readily evaluate the adequacy of their current configuration practices. Indicators that current CM practices are adequate:

- ➤ Ease in finding and identifying *all the necessary and sufficient* source code during the Year 2000 assessment phase; and
- ➤ Ability to readily create a definitive, complete listing of application components.

Under normal circumstances the procedures followed by the development teams are probably adequate to administer the system components and avoid accidentally overlaying a good component with an earlier version or overlooking a component. As the amount of change grows in response to the breadth and depth of Year 2000 activity, we will begin to see the signs of strained resources, with discipline breaking down. Pragmatically, it is very unlikely that we will be able to completely avoid parallel development (two teams working to introduce different requirements to the same code modules concurrently). This will strain less sophisticated methods of configuration management even further.

There are certain practices that should be considered for immediate implementation to facilitate addressing this problem:

- ➤ Implement effective configuration management techniques. See **Chapter 3**.
- ➤ Establish Libraries to isolate source and object code. Consider establishing the following libraries:
  - Production source,
  - Production object,
  - Year 2000 Development,
  - Maintenance,
  - Acceptance Year 2000 development,
  - Acceptance maintenance,
  - Staging to certification, and
  - Staging to production.
- ➤ Confirm that the source and objects match the production object. Recompile source, and relink. Compare the results of identical test scenarios processed against both the current production and the recompiled version. They should match.
- ➤ Add procedures to prevent accidental introduction of non-compliant code into the environment. Make it a conscious decision and routinize it.

- Add controls to prevent circumventing your procedure.
- Consider freezing production source and object libraries. Freezing a library means there is no way to place software into these libraries without appropriate authorization. This is a good way, procedurally, to make certain all the checks and reviews have taken place to avoid "reversing out" good software by accidentally installing a product that was not reconciled with the current production version. Testing libraries for final acceptance testing should also be frozen, and the approved code migrated into production only from the acceptance library. This way we are assured that what was tested is what actually wound up in production

On the **Hitachi** platform, automated configuration management support is provided by Computer Associates' Endevor product. A companion tool, Computer Associates' Parallel Development Manager, compares source code line by line in order to reconcile two versions of code that originated from the same parent but underwent two different revisions.

For the **Unisys** platform, SQA from Arkdata AB provides configuration management support that is similar to Endevor.

On the **PC/LAN**, Source Safe provides some library management capabilities for source code. PVCS can also be useful in this area.

1. 2. 4. The Testing and Certification Phase

The objective of system, regression and integration testing phases and of certification testing is to thwart any possibility of catastrophic failure by extensively certifying the quality of the changes, especially as the data is exchanged among databases, modules and applications, and as the data is interpreted and used across systems and organizational boundaries. The goal is to complete testing and certification by January 31, 1999.

Let's make it clear that there is no real way to guarantee that we found all the bugs or all the occasions when date functions will disrupt the functioning of an application.

The approach is to thoroughly test the application against the production version (a parallel test) to ensure the Year 2000 changes have not degraded the functional performance of the application, and then to advance the date into the next century to thoroughly confirm that the application carries out the date functions correctly.

A Year 2000 perspective pertaining to each of the above is provided in the following sections. Additional technical materials on Testing can be found in **Chapter 5**.

1. 2. 4. 1. Confirm that Changes did not Adversely Affect Current System

Perform a parallel test by running the same data through both the current production version and the renovated system and compare the results. There should be no differences if only Year 2000 changes were made to the software. If functional changes were introduced at the same time as the Year 2000 renovations, the parallel test results will not be identical. There will be differences that are the result of the functional changes introduced, and these differences can be "explained" based on the functional changes. A purely functional test, similar to that performed for a major release, may be more thorough than a parallel test, but the parallel test is more efficient. A parallel test may not require extensive functional expertise if there is a high confidence that the test scripts are comprehensive and fully exercise a broad scope of transactions and conditions.

For very critical applications with low functional confidence, there are specialized tools (test data generators and test coverage monitors) in the industry to help develop a comprehensive test bed.

1. 2. 4. 2. Confirm that the Application Continues to Perform Correctly with Future Dates

It is not enough to confirm that the renovated application functions for the current year. Future date testing is crucial. Adequate testing must simulate dates in the next century. Develop test data and scenarios appropriate for transactions the system is likely to experience during typical processing during the Year 2000 and 2001.

Experts encourage use of a parallel test approach as the most efficient means of addressing simulated future date testing. A parallel test simulating a future date requires that the test bed and transactions be "aged" and the "system" date advanced by the same amount.

If the application uses conditional logic based on day-of-week, selecting a twenty-eight year interval would be an advantage, since the day of the year falls on the same day of the week every 28 years.

File-Aid is a testing tool that supports changing and assessment of files on the Hitachi. To simulate different dates on that platform, use Simulate 2000. MODYDATE has been used for date simulation on the Unisys platform, although a new tool, Accommodate, is currently being evaluated to replace it. Date simulation on the PC/LAN is very application specific; there is no overall tool for that task. The same can be said for the task of changing and assessment of files on the PC/LAN. For more information, refer to the **HUD Year 2000 Tools Overview**, **Document F** in the **Reference Library**.

1. 2. 4. 3. Confirm that Boundary Dates can be Handled

Check specific future dates or intervals to assure that the software will continue its proper behavior. These boundary conditions include starting on December 31, 1999, and extending into January 1, 2000; February 28, 2000 and extending into "the next day"; February 29, 2000 and March 1, 2000. Additionally, there are known values in date fields that have, traditionally, held special meaning. We need to confirm that the system will behave properly as the calendar achieves each of these dates, such as 12/31/99 or 9/9/99.
To simulate different dates, use Simulate 2000 on the Hitachi. A date simulation package for Unisys is described in the **HUD Year 2000 Technical Bulletins**, **Document C** in the **Reference Library**. Such a tool is not needed on the PC/LAN.

1. 2. 4. 4. Certification

A system will be certified if and only if it has successfully completed current and future date testing in a fully compliant environment. Even after it has been certified, future modifications could be a victim of old patterns and behaviors, reducing a compliant system to a non-compliant state. Certification, therefore, is not and cannot be independent testing conducted as of a point in time. Certification is the convincing demonstration to a third party that all reasonable and prudent actions and controls have been exercised to predict with a high degree of certainty the integrity and correct performance of this software as it enters the time horizon of the Year 2000. Certification requires, therefore, a demonstration that adequate testing simulating future dates, boundaries and dates with special significance has occurred on a compliant platform, and further, that procedural controls and retesting are in place to inhibit contamination of a previously compliant system.

The means by which this is done will differ depending on whether we're addressing:

- Certification of renovated software;
- Certification of commercial off-the-shelf software; or
- Certification of newly developed software, built compliant.

In all cases we will want the application to demonstrate that it has successfully performed future date and boundary & special date testing on a fully compliant platform.

Further information on certification can be found in **Chapter 6**.

1. 2. 4. 5. The Implementation Phase

The objective of this phase is to address all issues associated with planning and executing the transition into production, including conversion, disaster recovery and archived data.

Since not all system components will be converted or replaced simultaneously, HUD will be operating in a heterogeneous computing environment comprised of a mix of Year 2000 compliant and non-compliant applications and system components. The reintegration of compliant applications and components into the production environment must be carefully coordinated to account for system interdependencies. In some cases, parallel processing—where the old and converted systems are run concurrently for a time—may be needed to reduce the risk.

To adequately prepare for implementation, the renovation team should:

- Define transition environment and procedures;
- Develop implementation schedule, including extensive contingency planning;
- Notify users;
- Resolve data exchange issues and external exchange concerns;
- Complete testing;
- Perform database and archive conversion;
- Update business resumption plans; and
- Move code into production.

Refer also to **Chapter 7** (Implementation).

(This page has been left blank intentionally.)